

## LA-UR-21-28707

Approved for public release; distribution is unlimited.

Title: Summary of GNU Octave Lady Godiva Simulator

Author(s): Kimpland, Robert Herbert  
Determan, John C.  
Kim, Seung Jun  
Wass, Alexander Joseph

Intended for: Report

Issued: 2021-09-01

---

**Disclaimer:**

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

## Summary of GNU Octave Lady Godiva Reactor Simulator

8/17/21

Robert Kimpland, John Determan, Jun Kim, Alexander Wass

### Introduction

A pseudo real-time interactive reactor simulator for the Lady Godiva reactor has been developed using the GNU Octave software package. The simulator tracks the time-dependent power and fuel temperature behavior of the Lady Godiva reactor during sub-prompt critical transients. Users interact with the simulator by controlling reactivity input. This input occurs through users repositioning a control rod as desired during simulator operation. Various reactor parameters are displayed graphically in near “real time,” facilitating user assessment and feedback input. The reactor model consists of a lumped-fuel core, a single control rod, and a simple heat transfer mechanism that allows heat to be rejected from the core. The goal of the simulator is to demonstrate in real-time the system response to reactivity perturbations, the effect of reactivity feedback, and the effect of temperature-dependent equilibrium conditions. The dynamic behavior of the reactor is primarily driven by the relative characteristic time-constants between the delayed neutrons and the overall heat transfer parameter. The simulator will allow users to gain experience performing standard reactor operations, such as establishing steady-state at different fuel temperatures and making power level changes.

### Lady Godiva Reactor Model

A simple dynamic model has been developed to simulate the transient behavior of the Lady Godiva Assembly. The goal of this effort is to test a suitable mathematical engine that powers potential reactor simulator platforms. This model combines the point reactor kinetics model with an energy equation for the reactor fuel, and a simple model that describes reactivity input from a control rod. These models are coupled together through a governing assembly reactivity equation and through energy deposition in the fuel. The assembly prompt neutron kinetics is described by the following expression

$$\frac{dP}{dt} = \frac{\beta}{\Lambda} \left[ (R - 1)P + \sum_i f_i D_i \right]$$

where

$$f_i = \frac{\beta_i}{\beta}$$

The delayed neutron precursor contribution is given by the following

$$\frac{dD_i}{dt} = \lambda_i(P - D_i)$$

The reactivity equation for the assembly is given by

$$R = R_{CR} + \alpha_T \Delta T_f$$

This equation includes a single reactivity feedback mechanism driven by fuel temperature. During reactor simulator operation, time-dependent reactivity input is controlled through user determined control rod positioning. The relationship between control rod position and assembly reactivity is given by

$$R_{CR} = -1 + \frac{2}{\pi} R_{CR,max} \left( \frac{\pi h}{2a} - \frac{1}{4} \sin \left( \frac{2\pi h}{a} \right) \right)$$

This expression is based on the one-group perturbation model for a simple control rod. The reactivity input to the assembly by the movement of the control rod is a function of the total integral reactivity worth of the control rod  $R_{CR,max}$ , and its position  $h$ , within the core.

During reactor simulator operation, fission energy is deposited into the fuel causing its temperature to increase. As the fuel temperature increases, heat transfer via natural convection occurs, causing heat to be rejected from the assembly. This energy balance scheme allows for steady-state operating conditions to be established at various excess reactivity inputs. The temperature model for the fuel is given by

$$\frac{dT_f}{dt} = \frac{p}{M_f C_p} - \gamma(T_f - T_o)$$

where the assembly power is given by

$$p = p_o P$$

## Index

$a$  = Active length of Control Rod (6 inches)

$\alpha_T$  = Temperature coefficient of reactivity feedback ( $\$/K$ )

$\beta$  = Total delayed neutron fraction

$\beta_i$  = Delayed neutron fraction of  $i$ th group

$C_p$  = Specific heat of fuel (MJ/kg/K)

$D_i$  = Normalized delayed neutron precursor contribution of  $i$ th group

$f_i$  = Normalized delayed neutron fraction of  $i$ th group

$\gamma$  = Overall heat transfer coefficient (1/s)

$h$  = Position of Control Rod (inches)

$\Lambda$  = Mean neutron generation time (s)

$\lambda_i$  = Delayed neutron decay constant for  $i$ th group (1/s)

$M_f$  = Mass of fuel (kg)

$P$  = Normalized assembly power

$p$  = Assembly power (MW)

$p_o$  = Initial assembly power (MW)

$R$  = Assembly reactivity ( $\$$ )

$R_{CR}$  = Control Rod reactivity input ( $\$$ )

$R_{CR,max}$  = Total Control Rod integral reactivity worth (1.40  $\$$ )

$T_f$  = Assembly fuel temperature (K)

$T_o$  = Initial assembly fuel temperature (K)

$t$  = time (s)

## GNU Octave Lady Godiva Reactor Simulator Model

The LadyGRS55.m file contains the complete stand-alone Octave reactor simulator model. The simulator is driven by the MATLAB ODE solver ode23s, which employs the anonymous function f\_anon to combine the functions eval\_rhs and eval\_input. The function eval\_rhs contains the state variable definitions for the Lady Godiva reactor model. The function eval\_input contains the dependent variable definitions for the Lady Godiva reactivity model.

The “real time” nature of the simulator is based on an execution scheme using the Octave tic, toc, and pause functions. Within a for-loop, the tic function is initiated and the user controlled reactivity input is interrogated. The ODE solver then solves a 10 second window of the reactor model. The results of the ODE solver are concatenated each cycle, and the 10 second results and the total running results are reported. After these tasks are performed, the toc function is called to obtain the “wall clock” elapsed time. The pause function is then used to allow 10 seconds of wall clock minus the cycle execution elapsed time to pass before starting the next cycle.

The simulator begins by displaying a title and the current operating conditions at the Octave command line. The Lady Godiva reactivity model is pre-set to have delayed critical occur at a control rod position of approximately 3.67” while the fuel temperature remains at the initial fuel temperature. The term zero-power, which is displayed along with the control rod position, implies an initial starting assembly power that is sufficiently low to keep the fuel temperature roughly constant at the beginning of a simulation. The simulator then generates an integral control rod worth curve using the function controlrod2, and displays the data in Figure 1. This curve relates the reactivity worth of the control rod to its position within the assembly. The user is then prompted to enter the initial starting power of the assembly and the initial fuel temperature at the Octave command line. The Octave input function is used to query these values. With the initial fuel temperature obtained, the initial conditions for the state variables are defined in the 9x1 array x0.

At this point, an Octave for-loop is initiated that drives the cyclical ODE solver scheme. The tic function is called to start the cycle’s running clock. The user controlled control rod position, which is located in a text file called CRP.txt, is automatically interrogated by the simulator through the use of the Octave readtext function. This interrogation occurs every cycle whether the user chooses to make an input change or not. An Octave if-statement checks the value of the control rod position to see if a simulator termination flag has been given by the user. A control rod position greater than 6” signals simulator termination and the break function ends the for-loop execution. The control rod position input is converted to reactivity with the function controlrod. Next the 6x1 time-dependent reactivity input array RE is defined.

The tin and tout parameters specify the time boundaries for each successive 10 second window. Each cycle they are updated and sent to the ODE solver through the ts array. The anonymous function f\_anon is defined and the ode23s solver is called. The ODE solver generates a 10x9 matrix called xs that contains the output state variables for the 10 second window. The matrix gs, which contains the running tally of state variables through every cycle, is updated by using the matrix concatenation operator, gs=[gs;xs]. Likewise, the array tg, which contains the running tally of times through every cycle, is updated using the concatenation operator, tg=[tg;ts].

With the output matrices `xs` and `gs` updated, Figure 2 is generated. The reactor power and fuel temperature are displayed graphically for both the latest 10 second window and the aggregate running time. The time integrated power is also displayed along with the current control rod position.

The state variable initial conditions array `x0`, is updated by redefining it as `x0=xs(10,:)`. The next cycle's initial conditions are the final values of the state variable output matrix `xs`, note the re-orientation of the array from a row to a column. This step pieces each successive 10 second window together allowing the continuous solution of the set of model ODEs through time with user input interactions. Finally, with all the cycle tasks completed, the cycle's elapsed execution time is found by calling the `toc` function. The `pause` function is used to suspend the simulator execution, so that exactly 10 seconds of wall clock passes between each solver cycle. This pause creates the "real time" nature of the simulator since 10 seconds of wall clock time produces 10 seconds of model ODE solution results. At this point, the Octave `endfor` statement signals the beginning of a new cycle.

### **GUI Installation/Operating Instructions**

For installation on Windows 10:

- 1.) Download R, install.

**<https://cran.r-project.org/bin/windows/base/>**

You may see a message about needing administrative rights, and even error messages, near the end of the install, but these can be ignored. It is best to not use admin rights during this install. Also, modify the default install location. The default path should be `<user home folder>/Documents/R/R-4.0.x` – omit "Documents" from this path, yielding:

`<user home folder>/R/R-4.0.x`

as the R installation directory.

R: R-4.0.x-win.exe (x was 4 when I started, and 5 at this writing)

- 2.) Download R Studio, install. If prompted to provide an admin password, do so.

**<https://www.rstudio.com/products/rstudio/download/#download>**

RStudio: RStudio-1.4.1106.exez

- 3.) Download and install Rtools, for building some packages on windows:

**<http://cran.rstudio.com/bin/windows/Rtools/>**

download 32 or 64 bit version as appropriate, and install.

- 4.) R will call Python and Python will call the Octave “.m” file reactor model, so install Gnu Octave at this point if not already installed. Download and run windows installer from [gnu.org](http://gnu.org).

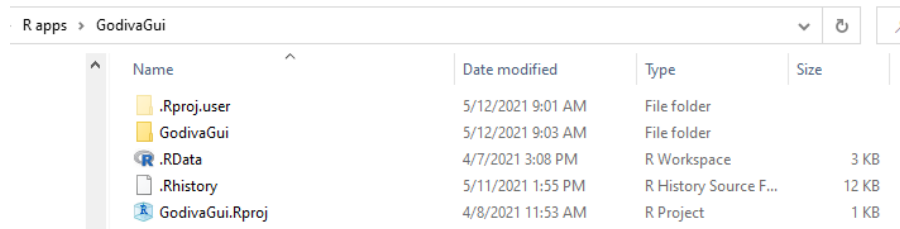
The user should verify paths on their system, in the discussion that follows.

Set the windows environment PATH variable to include the path to octave-cli.exe – by default octave-cli should install to C:\<user home folder>\AppData\Local\Programs\GNU Octave\Octave-x.x.x\mingw64\bin, but you can search on the executable name to find it. To set the environment variable, type “environment” in the windows search box and click on the entry “Edit the system environment variables”. In the resulting dialog click “Environment Variables...”. Under System Variables, find “Path” and double click it. Click new and type or paste in the path to octave-cli.exe.

We also need to set a path for the tools in Rtools. Click “New” again and enter “C:\rtools40\usr\bin”.

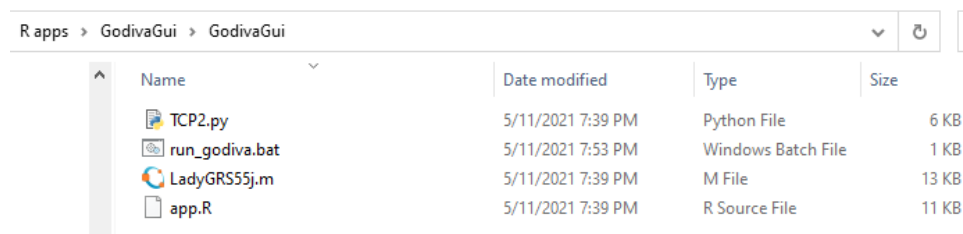
Click OK 3 times to exit the dialogs. If R is already running, it is like best to stop and restart R Studio to be sure it is aware of the new path information.

- 5.) Put files from our delivery into a path for R projects – for simplicity I suggest using c:/R Apps/ as the base directory. Create folder “R Apps/” at C: and extract zip file contents to this directory. Make sure that the file GodivaGui.Rproj reside sat C:/R Apps/GodivaGui, as shown below:



Name	Date modified	Type	Size
.Rproj.user	5/12/2021 9:01 AM	File folder	
GodivaGui	5/12/2021 9:03 AM	File folder	
.RData	4/7/2021 3:08 PM	R Workspace	3 KB
.Rhistory	5/11/2021 1:55 PM	R History Source F...	12 KB
GodivaGui.Rproj	4/8/2021 11:53 AM	R Project	1 KB

and C:/R Apps/GodivaGui/GodivaGui should look like:



Name	Date modified	Type	Size
TCP2.py	5/11/2021 7:39 PM	Python File	6 KB
run_godiva.bat	5/11/2021 7:53 PM	Windows Batch File	1 KB
LadyGRS55j.m	5/11/2021 7:39 PM	M File	13 KB
app.R	5/11/2021 7:39 PM	R Source File	11 KB

- 6.) Open R Studio. In R Studio , click File / Open Project... . In the resulting dialog, navigate to C:/R Apps/GodivaGui and select GodivaGui.Rproj. The rest of the work will be done by installing packages into R, much as is done in Octave.
- 7.) Go to R console window. R needs to know some path information to use Rtools. From the R console type (typing is best, copying from these instructions produces inappropriate quotation



characters, and errors):

```
writeLines('PATH="{RTOOLS40_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviro")
```

This will create a file in your Documents directory “.Renviro” containing the path to Rtools.

If you are behind a firewall you will get an error indicating you can’t access the internet, you must also set the proxy url in the .Renviro file to:

```
http_proxy=<proxy string for your location>  
or  
https_proxy=<proxy string for your location>
```

But fair warning, I was generally not behind our firewall, and I have not worked out all the details of this issue, I just know I ran into the error when I was behind our firewall.

Exit and restart RStudio to read this file.

In the R console type:

```
Sys.which("make")
```

Since we set the rtools path earlier, you should see the path we set at that time

#### 8.) Some general notes on handling errors during R package installation:

- a) make sure you have a connection to the internet
- b) try repeating a failed command
- c) close R, restart R, and try again, If you performed a “library(<package name>)” command, you will need to do that again for the new R session.

These steps, as needed will generally get you through errors in package installation.

In the console, type

```
install.packages("shiny")
```

Say “yes” to the prompt.

#### 9.) We need to use python to be able to run the Octave “.m” file, so we load the “reticulate” library which integrates python into R.

In the R console type:

```
install.packages("reticulate")
```

When this finishes we can test reticulate by typing, in the R Console:

```
library("reticulate")          # library loads packages for use
```

```
ocpy <- import("oct2py")      # ocpy is just a name
```

What you will likely see at this point is a message saying R Studio does not know about a python installation. While you can use reticulate commands to specify an existing python interpreter, I would suggest saying "Y" to the question R Studio is asking – "Would you like to install Miniconda? [Y/n]" This is a pared down version of Python, and sufficient for our need. Preparing instructions to deal with other possible Python installations is beyond the scope of these instructions.

Type "Y" to install Miniconda; at the end of the installation there will be an error message about not being able to install "oct2py", but this is normal. After installation finishes, we need to install "oct2py" into miniconda – reticulate gives us `py_install` for acquiring python packages.

Type:

```
py_install("oct2py")
```

Type:

```
library("reticulate") # you can also use up arrows, even after exiting and restarting R)
```

Repeat the "`ocpy <- ...`" command (up arrow to retrieve command, like most consoles)

Type:

```
ocpy
```

If all is well, the system will respond by printing what ocpy is:

```
Module(oct2py)
```

10.) Verify that you have a Python installation.

Open the Environmental Variables dialog again (see step 4 above). Set a path to the miniconda package that we just installed in step 9. Click "New" and enter:

C:\<user home folder>\AppData\Local\r-miniconda

Click OK 3 times to exit the dialogs. If R is already running, it is like best to stop and restart R Studio to be sure it is aware of the new path information.

Open a cmd window, type python – if a python interpreter opens, all is well. Type “quit()” to exit the interpreter.

- 11.) Let's install a few more R libraries that will be used, for plotting and string manipulation. It is best to close and reopen R Studio before continuing. If any of the following packages fails to load, the first remedy is to close and reopen R and retry the failed installation.

In the R console type:

```
install.packages("stringr" )
```

Say “yes” to the prompt.

- 12.) In the R console type:

```
install.packages("utf8")
```

- 13.) In the R console type:

```
install.packages("ggplot2")
```

- 14.) In the R console type:

```
install.packages("gridExtra")
```

- 15.) In the R console type:

```
install.packages("dplyr")
```

- 16.) To see the results of this process, in the Windows search window, type:

cmd

A command window will open. Change directory to the C:/R Apps/GodivaGui/GodivaGui folder.  
Type:

```
run_godiva <path to R Studio on your machine>
```

The path should be <user home folder>/R/R-4.0.x/bin.

Two command windows will open – one to run the TCP server, and the other to execute the R Shiny application. Click the “Start” button. Some time may elapse, so be patient, but you should see the graphs open on the right side of the window, eventually. Once the graphs are present, use the slider arrows to change the control rod position and watch the graphs eventually respond (10 – 20 s). Click the “Stop” button to end the application.